TR-86-002

Automatic Curve Fitting with Quadratic B-spline Functions and Its Application to Computer-assisted Animation

Mark C. K. Yang[*], Chong-Kyo Kim[**], Kuo-Young Cheng[+], Chung-Chun Yang[++] and S. S. Liu[+]

[*] Department of Statistics, University of Florida, Gainesville, Florida 32601

[**] Department of Electronic Engineering, Chonpuk National University, Korea, ROK

[+] Institute of Information Science, Academia Sinica, Taiwan, ROC

[++] Naval Research Laboratory, Washington, D. C. 20375

November 18, 1985

Dr. M.K.C. Yang
Code 5381
Naval Research Lab
Washington, D.C. 20375-5000

Dear Dr. Yang:

I am happy to inform you that your manuscript, "Automatic Curve Fitting with Quadratic B-spline Functions and Its Applications to Computer-assisted Animation", has been accepted for publication in CVGIP subject to editing. You should be receiving galley proofs and a reprint order form about two months before the paper is published.

Figures, proofs, and any correspondence about the status of the paper should be sent directly to the publisher (Attention: CVGIP, Journal Publishing Department). Correspondence involving changes in the text of the paper (prior to the galley-proof stage) should be sent to me, not to the publisher. In either case, please reference Manuscript No. 2097.

Thank you for your time.

Cordially,

K. Logan

Kathleen E. Logan
Editorial Assistant

kl

Enclosures

Abstract

Automatic fitting to digitized curve by quadratic B-spline functions is discussed. Due to its simplicity, it is possible to find a quick fit within a given error tolerance bound which is defined as the maximum distance between the fitted curve and the original curve. Again due to the simple form of a quadratic polynomial, some complicated computations in computer-assisted animation such as hidden line removal can also be easily handled.

Keywords: Computer graphics, B-spline function, computer-assisted animation, curve fitting.

Most of this research was done while the first four authors were visiting the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, Illinois.

1. Introduction

Representing curves by B-spline functions was suggested by Riesenfeld [14] and has been found to be very useful in many areas such as computer-aided design, computer graphics, and pattern recognition. In particular, a B-spline function represented by guiding points can save a tremendous amount of memory for curve storage and generation. There are quite a few available methods for constructing a B-spline function to fit a given curve. However, most of them are concentrated on a cubic spline function. For example, Wu, Abel, and Greenberg [16] developed an interactive system to fit a curve by a cubic spline. They used the basic relation between knot points $P_0, P_1, \ldots, P_m$ and their guiding points $V_0, V_1, \ldots, V_m$

$$\frac{1}{6}\begin{pmatrix} 4 & 1 & 0 & 0 & \ldots & 0 \\ 1 & 4 & 1 & 0 & \ldots & 0 \\ & & \ldots & & & \\ 0 & & & 1 & 4 & 1 \\ 1 & & & 0 & 1 & 4 \end{pmatrix}\begin{pmatrix} V_0 \\ V_1 \\ . \\ V_{m-1} \\ V_m \end{pmatrix} = \begin{pmatrix} P_0 \\ P_1 \\ . \\ P_{m-1} \\ P_m \end{pmatrix} \qquad (1)$$

for a closed curve and

$$\frac{1}{6}\begin{pmatrix} 6 & 0 & 0 & \ldots & & 0 \\ 1 & 4 & 1 & \ldots & & 0 \\ & & \ldots\ldots & & & \\ 0 & 0 & 0 & \ldots 1 & 4 & 1 \\ 0 & \ldots & & 0 & 0 & 6 \end{pmatrix}\begin{pmatrix} V_0 \\ V_1 \\ . \\ V_{m-1} \\ V_m \end{pmatrix} = \begin{pmatrix} P_0 \\ P_1 \\ . \\ P_{m-1} \\ P_m \end{pmatrix} \qquad (2)$$

for an open curve. The derivation of (1) and (2) and some details of a cubic spline function can be found in a recent book by Pavlidis [12]. Yamaguchi [17] found an iterative method to solve (1) and (2) for large m and his method was later adapted by Lozover and Preiss [10] to develop an automatic scheme for cubic B-spline generation. However, no error analysis between the original curve and the fitted one was discussed. Hence their method is not fully automatic given a required error tolerance. Although a quadratic B-spline function is simpler and almost as smooth as a cubic spline to the naked eyes (see for example, Fig. 11.8 of Pavlidis [12] and Fig. 3 of Wu, et al. [16]), their constructions seem lacking in the literature. Part of the reason would be that the formulas corresponding to (1) and (2) for a quadratic B-spline are generally unsolvable. If one follows the relation between the guiding points and knot points of a quadratic spline, then (1) becomes

$$\frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & \cdots & 1 \\ 1 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 1 & & 0 \\ & & \cdots & & \\ 0 & & & 1 & 1 \end{pmatrix}\begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ \cdot \\ V_m \end{pmatrix} = \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ \cdot \\ P_m \end{pmatrix} .$$

This matrix becomes singular for m > 3. The first part of this paper is to fill this gap by a different approach.

One of the main advantages of using a quadratic B-spline representation over a cubic one is that in quadratic B-spline the positions of the actual curve is much closer to the guiding points (see Fig. 8.9 of Ballard and Brown [1]). Hence it is easier to pinpoint the curve's position by its quadratic spline guiding points. Also, other parameters of a curve such as the enclosed area of a closed curve or the perimeter length of a curve are much easier to compute in a quadratic spline representation. For example, the curve length of a quadratic B-spline can be represented by an elementary function of the coordinates of its guiding points (see Appendix F), while that for a cubic spline numerical integration seems to be the only solution. Similarly, for hidden line removal , the intersection of two quadratic splines can be examined by analytic formula (see Appendix E), while the intersection of two cubic spline functions can only be done by iterative procedures.

One important reason for representing a given curve by a simple function is for easy picture alternation. Computer-assisted animation is a perfect example. There have been several systems in computer animation (see Burtnyk and Wein [3]), but none of them used B-spline function for curve representation. Part of the reason again seems to be that most literature on B-spline functons has been concentrated on the cubic B-spline function which makes some computations in animation such as the hidden line removal and the unequal guiding point matching very time consuming. The computation is much simpler in the case of quadratic B-splines. Cartoons developed by this method were shown to school children and it was considered to be good from their point of view. However, it is not the intention of this paper to solve the artistic problems stated in Catmull [4] or some other problems such as aliasing in rapid movements stated in Korein and Badler [7]. We feel that most of these types of problems are basically trade-offs between the animator's time and the output quality of the product. They are usually hard to quantify. Rather, we emphasize on the feasibility of this new method which has provided a new avenue in computer-assisted animation design.

2. Notations and methods of curve fitting

In this paper we use (x, y) to denote a point in the plane. If, $G_i = (x_i, y_i)$, i=1,2,...,m are m consecutive guiding points of a quadratic B-spline function, then it is well known (e.g. Pavlidis [12], p.266) that the curve generated by these guiding points are piecewise parabolas passing through the middle points of the neighboring guiding points. More precisely, let

$$P_i = (G_{i-1} + G_i)/2, \quad i=1,2,...,m. \tag{3}$$

Then the quadratic B-spline function will pass through all the P's and tangent to the polygon formed by the adjacent guiding points. The piecewise parabola between P_i and P_{i-1} is

$$C_i(u) = \{(G_{i-1} - 2G_i + G_{i+1})u^2 + 2(G_i - G_{i-1})u + (G_i + G_{i-1})\}/2, \tag{4}$$

where $u \in [0,1]$. Apparently (4) is easier to implement and it also has an intuitive appearing in curve generating. (see Chaikin [5] and Riesenfeld [15]).

The contour fitting process is divided into two basic steps. Each step is given an error tolerance ε_i, i=1,2. Let the original digitized contour be listed consecutively as

$$e_i = (x_i, y_{i-1}), \quad i=1,2,...n, \tag{5}$$

and the fitted spline curve be S. Then we wish for any i=1,2,...,n,

$$||S - e_i|| < \varepsilon_1 + \varepsilon_2$$

where $||S - e_i||$ denote the distance between point e_i and curve S, i.e.

$$||S - e_i|| = \min_{z \in S} d(z, e_i)$$

and $d(z,e_i)$ denotes the Euclidean distance between a point $z \varepsilon S$ and the point e_i. Since the user may require only $||S - e_i||<\varepsilon$ for some ε, a natural choice of ε_1 and ε_2 is $\varepsilon_1 = \varepsilon_2 = \varepsilon/2$.

In the first step, parabolas are generated to fit the points e_i, $i=1,...,n$. Note that a parabola is a special case of a conic curve, and its construction has been studied by many authors. Bookstein [2] derives a least squares method for conic curve fitting, Liao [8] uses a two stage procedure first by fitting the digitized contour with a polygon and then fitting the polygon with a conic curve by the least squares method, and Pavlidis [13] gives an intensive discussion of the relation between a fitted conic curve and its guiding polygon. However, none of these methods discussed the exact distance between the fitted curve and the original data points. Bookstein's and Liao's methods are not suitable for our purpose because for given knot points and the continuity requirement of tangents, the parabolas are uniquely determined. The fitting method by Pavlidis [13] can be adapted here. We modify it as follows:

Let m_1 and m_2 denote respectively the slopes at points $z_1 = (x_1, y_1)$ and $z_2 = (x_2, y_2)$ and the equation of the fitted parabola be expressed as

$$(x + By)^2 + Dx + Ey + F=0. \qquad (6)$$

The values of B, D, E, and F are given in Appendix A. Formula (6) is not as convenient as (4) for curve generation, but it is easier to compute the distances as required in the rest of our procedure.

The choice of knot points can be done sequentially. For a closed curve the first knot can be arbitrarily chosen, but for an open curve it is the first data point. The next knot will be picked starting at the third data point from the first knot. The slope at each knot will be the average of the two neighboring slopes, i.e. for points $A=(x_1,y_1)$, $B=(x_2,y_2)$, $C=(x_3,y_3)$, the slope at the center point B is

$$m = \frac{1}{2}\left(\frac{y_2 - y_1}{x_2 - x_1} + \frac{y_3 - y_2}{x_3 - x_2}\right) \qquad (8)$$

and if the knot point is the first or the last point of an open curve, then the slope will be computed from one side only. For example, if A is the starting point, then the slope at A is

$$m = \frac{y_2 - y_1}{x_2 - x_1} \quad . \qquad (9)$$

Sometimes it would be important and necessary to preserve some sharp turns of a curve at a cusp point by using a double guiding points (see Pavlidis [12], p 268). In this case the tangents at both sides of this knot will not be the same. To what extent a turning angle is considered as sharp enough for double guiding points has to be determined by the user. When the angle at B, computed by the difference between the arctangents of the two slopes at both sides of B, is larger than a given threshold, double guiding points will be used.

The fitness of each parabola will be measured by the maximum distance between this parabola and all the points in (5) it represents. The computation of this distance is described in Appendix B. If this maximum distance is smaller than ε_1, the next data point will be picked as a new knot and the above process will be repeated until the fitting error is greater than ε_1, then the previous point will be considered as the end knot of the current fitting and the starting knot of the next piece of parabola.

Now suppose we have fitted the digitized curve with piecewise parabola P_i , $i=1,2,\ldots,m$ and each parabola is represented by the general form (6). Let the coefficients of the ith parabola be (B_i , D_i , E_i , F_i). Then the present step is to fit these parabolas by a B-spline satisfying

$$\max \|S - P\| < \varepsilon_2 ,$$

where S represents the B-spline and P represents the parabolas, and max means the maximum distance between S and P. This can be done by the following steps.

1. Find the intersections of all the tangents at the knot points and these intersections will be used as the initial guiding points. Because two tangents of a parabola can never be parallel, the intersections are guaranteed.

2. Find the maximum distance between the fitted spline and the parabola according to the procedure described in Appendix C.

3. If the maximum distance is smaller than ε_2 , the process should stop. Otherwise add one more guiding point to the existing guiding points and go back to step 2. The procedure of adding a new guiding point to a spline function as well as the proof of its convergence are given in Appendix D.

3. An Example in Curve Fitting

Since the parabola fitting is quite straightforward, we will discuss our example starting from the second step.

Figure 1 is a curve that is composed of 11 parabolas. The joints of these parabolas are Q_1, $Q_2,\ldots,$ Q_{11} and the coordinates and slopes at these points are given in Table 1.

According to the step 1 of the guiding points finding procedure, the initial points are the intersections of the tangents at these knot points. They are shown as G_1 (i,1) in Table 1. The notation G_k (i,j) denotes the jth guiding point for parabola i at the kth iteration.

When G_1 (i,1) are used as guiding points, the B-spline function produces a maximum discrepancy of 1.696 at the 11th parabola (see Table 2 1st iteration column). Suppose $\varepsilon_2 < 1.696$. Then the guiding points of parabola 11 needs to be modified. Using the procedure described in Appendix D, we found the new guiding points for parabola 11 to be;

$$G_2(11,1)=(6.296,15.681),\ G_2(11,2)=(8.1,11.3).$$

From Table 2 we see that the largest discrepancy now is 0.87 at parabola 9. If the error tolerance ε_2 is 1.0, then the 12 guiding points are now acceptable for the spline approximation. The original parabolas and their present fit are shown in Figure 2. When this process is continued once more the maximum discrepancy becomes 0.347 at parabola 7. The spline for $\varepsilon_2 = 0.2$ is achieved at the 8th iteration. The last two iterations are given in Table 3 and the fit at the 8th iteration is given in figure 3.

Now we shall illustrate how a quadratic B-spline can be used in animation.

4. General 2-D Animation Hierarchical System

We would like to describe a general 2-D computer-aided animation system as a sequence of completely or partial drawn key frames which will be completed and interpolated by a computer. Without loss of generality, we need only to discuss the generation and the interpolation of two consecutive key frames. Suppose the first key frame is completely drawn and the second key frame is either completely drawn or specified by the user with some ideas of how it should differ from the first. From our experience of animation generation, three basic operations; shifting, rotation, and scaling are enough to describe the local changes of two consecutive frames. Moreover, we have found it convenient if the frames are described as a hierarchical system which will be illustrated by an example.

Let there be a tail wagging dog playing on a moving car. Then this picture can be represented by the hierarchical system in Figure 4. In this figure, the centers do not necessarily mean the geometric center of an element. They are the axes of rotations. For instance, the center of the tail should be at the body end of the tail. For any point, there are two coordinate systems ; one related to its center for the animator's convenience and the other is the screen coordinate system for display. The two coordinate systems are related in a very simple way. Suppose a point A has coordinate (x,y) with respect to its center O which has coordinate (x_0 , y_0) in the screen system. Then the coordinate of A is

$$(x_0 + x , y_0 + y) \tag{10}$$

with respect to the screen system. We define (x,y) as A's <u>local coordinate</u> and (10) as its <u>screen coordinate.</u>

Each curve or contour in the picture is assigned to a center in the hierarchical system and so is each center. For example, the center of the dog's body and the contour of the car, except its wheels, are controlled by the first hierarchy center 1 in Figure 4, while the main body of the dog, except the moving parts such as the tail, the ears, and the legs, is controlled by the second hierarchy center 3. The movement of each curve will be controlled by its center as well as a movement function. Let P(t) denote the local position of a point P at time t. In computer animation, we usually let t=0 for the first frame and t=1 for the second one. If n-1 intermediate frames are to be generated, then we need the coordinates of

$$P(t),\ t=i/n,\ i=1,2,\dots,n-1.$$

In order to describe the detail of P(t), we need a time function to describe scaling, shifting, and rotation for each curve. Let S(t) denote the local shifting of the center O(t), $\theta(t)$ denote the rotation angle, and $\alpha(t)$, $\alpha_0(t)$ denote the local scaling and the scaling proceeding to this hierarchy respectively. Then the new local coordinate with respect to its center O(t) is

$$P(t)= \alpha_0(t)[\ \alpha(t)A(t)A(0) + S(t)\ ]$$

$$A(t)=\begin{vmatrix} \cos\theta(t) & \sin\theta(t) \\ -\sin\theta(t) & \cos\theta(t) \end{vmatrix}, \qquad (11)$$

Again in Figure 4, suppose that the car moves with constant speed v (a vector) and away from the viewer. Then we have

$$S(t)=vt\ ,$$
$$\theta(t)=0\ ,$$
$$\alpha(t)= 1 - \xi t.$$

where ξ is the shrinking factor of the car because it is moving away from the viewer. Now suppose the dog will move and rotate according to

$$S_d(t) = v_d t,$$
$$\theta_d(t) = \omega t,$$
$$\alpha_d(t) = 1 - \xi_d t,$$

where v_d , ω , and ξ_d are self-explanatory. Then any point that is directly controlled by the dog at time t , according to (11), is

$$P(t) = (1-\xi t)[(1-\xi_d t)A(t)\ P(0) + v_d t]$$

$$A(t) = \begin{vmatrix} \cos\omega t & \sin\omega t \\ -\sin\omega t & \cos\omega t \end{vmatrix}.$$

Note that the center for P(t) is $(1-\xi t)[O_d + vt]$, where O_d is the center of the dog at time 0. Similar computation can be easily derived in the lower hierarchical level. A movement function needs not be linear. For example, Parke [11] has found that a cosine function can describe facial muscular movements better than a linear function in animation. However, for pure scaling and constant speed shifting, the implementation can be simplified, i.e. all the local movements can be omitted in the computation.

Theorem 1. If n-1 equal time intermediate frames are to be generated and all the movements in the hierarchical system involve only scaling and constant speed shifting, then a point, with screen coordinate (x_1 , y_1) in the first frame and (x_2 , y_2) in the second, has the following screen coordinate at time t,

$$(1-t)\binom{x_1}{y_1} + t\binom{x_2 - x_1}{y_2 - y_1}. \qquad (12)$$

The proof is quite easy. Moreover, (12) is a good approximation if the rotation angular velocity is small , i.e. $\omega \ll 1$.

The quadratic B-spline representation has some properties that are particularly important in animation.

1. It is easy to patch curves by the guiding points. For example, suppose the cartoonist wants the dog's tail to fly away from its body in the second frame. Then he has to patch the empty space on the body left open by the tail. A smooth patching, interactive with a computer, is not simple. But the B-spline function after losing the guiding points of the tail can patch the open space smoothly using only the guiding points of the body.

2. Hidden line removal can be solved by algebraic formula. Hidden line removal is one of the most cumbersome and time consuming procedures in computer animation. For example, when the dog's ears are flipping, it continually covers some, but different portions of its background. A general hidden removal algorithm for intersecting curves is usually complicated and slow (see e.g. Little and Peucker [9]). But with quadratic B-spline representation, the work is much easier. The detail is given in Appendix E.

5. Implementation and an Example

Since all curves are represented by the guiding points of a quadratic B-spline function, we will first show a relation between a changing curve and its guiding points.

Theorem 2. The animation transformation (11) to any curve in a quadratic B-spline function is the same as taking the transformation on the guiding points first and then generating the curve by the guiding points.

Proof. Let a curve C(u), 0 < u < 1, is guided by three guiding points G_1, G_2, and G_3. Then by (4),

$$C(u)=\frac{1}{2}u^2G_1+\left[\frac{3}{4}-(u-\frac{1}{2})^2\right]G_2+\frac{1}{2}(1-u)^2G_3, \quad 0<u<1. \quad (13)$$

Thus, the point $C_t(u)$ at time t is, by (11),

$$C_t(u)=\alpha_0(t)[\alpha(t)A(t)C(u)+S(t)] \quad (14)$$

Since

$$\frac{1}{2}u^2+\left[\frac{3}{4}-(u-\frac{1}{2})^2\right]+\frac{1}{2}(u-1)^2=1,$$

after substituting (13) to (14), we get

$$C_t(u)=\frac{1}{2}u^2G_1(t)+\left[\frac{3}{4}-(u-\frac{1}{2})^2\right]G_2(t)+\frac{1}{2}(u-1)^2G_3(t)$$

with

$$G_i(t)=\alpha_0(t)[\alpha(t)A(t)G_i+S(t)].$$

Hence, one quick way to generate intermediate pictures is to first find all the intermediate guiding points, then use (13) to generate the curve.

Sometimes a figure in key frame 1 may disappear in frame 2 or a figure not in frame 1 emerges in frame 2. Since the two problems are reciprocal, we consider only the former. A figure may disappear in the screen by shrinking to zero or by moving out of the screen from the edges. Both cases can be easily handled. A point can be represented as $\alpha(t)=0$ in (11). Thus if the curve is first transformed to a point and then the point is removed from the next frame, it has vanished. For a figure moving out from the edges, one needs to enlarge the frame and consider the screen as a window of the enlarged one.

Thus the problem of disappearing and emerging is solved.

Another more complicated problem appears when it is necessary to use unequal numbers of guiding points to represent the same curve at the two frames. The detail of this type of interpolation is given in Appendix F. Since considerable computing time is necessary in this case, unequal guiding points for corresponding consecutive curves should be avoided as much as possible. But since bending guiding points can richly change the shape of a curve, we seldomly found it necessary to use unequal guiding points to represent the same curve with slight variations.

Figure 5 is a simple animation picture. Only the first picture (1) is completely drawn and the other frame (4) is drawn from the hierarchical system in Fig. 6. All the other pictures are interpolations. The details of the implementation is given in Table 4. The hidden line removal procedure described in Appendix E is used in the ear, leg, and head movements. Parts of the body, bowl, and mouth are removed when necessary.

Appendix A. Fitting a parabola to two given points and slopes

To find the parabola that passes through two given points (x_1, y_1) and (x_2, y_2) with slopes m_1 and m_2 respectively at these two points is straight-forward. The B, D, E, F in (6) can be shown to be

$$B=-\frac{(y_2-y_1)(x_2^2-x_1^2)(m_2-m_1)+2(x_2-x_1)(x_2m_1-x_1m_2)-2(x_2-x_1)(y_2-y_1)}{(x_2-x_1)(y_2^2-y_1^2)(m_2-m_1)-2(y_2-y_1)(y_2m_1-y_1m_2)+2m_1m_2(y_2-y_1)(x_2-x_1)}$$

$$D=2[(x_2m_1-x_1m_2)+B[(y_2m_1-y_1m_2)+m_1m_1(x_2-x_1)]+B^2m_1m_2(y_2-y_1)\}/(m_2-m_1)$$

$$E=2\{(x_2-x_1)+B(y_2-y_1+x_2m_2-x_1m_1)+B^2(y_2m_1-y_1m_2)\}/(m_2-m_1)$$

$$F=-(x_1+By_1)^2-Dx_1-Ey_1 \ .$$

Appendix B. Finding the minimum distance between a point and a parabola

Let the point be (x_0, y_0) and the parabola be

$$(x+By)^2+Dx+Ey+F=0 \qquad (B.1)$$

The following transform will transfer (B.1) into the standard form $y'=x'^2$.

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \alpha\left(A\begin{pmatrix} x \\ y \end{pmatrix}+\begin{pmatrix} h \\ k \end{pmatrix}\right),$$

where

$$A = \begin{vmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{vmatrix} \qquad =\tan^{-1} B, \text{ and } |\theta| < \pi/2;$$

$h=d/(2a)$; $a=1+B^2$; $k=(F-h/2)/e$;

$\alpha=-a/e$;

$$\begin{pmatrix} d \\ e \end{pmatrix} = A \begin{pmatrix} D \\ E \end{pmatrix}. \qquad \text{(B.2)}$$

Let (x_0 ,y_0) be transferred to (x_0' , y_0') by (B.2). For simplicity, we retain the notations (x_0 ,y_0) and $y=x^2$. To find the minimum distance between (x_0, y_0) and $y=x^2$ is the same as to find an x such that $(x_0 - x)^2 +(y_0- x^2)^2$ is minimized. By differentiation, x should satisfy

$$2x^3 + (1-2y_0)x- x_0=0,$$

which can be solved by the root formula for a cubic equation.

Appendix C. Finding the maximum distance between the fitted quadratic B-spline and the original parabola

Let us use Figure C.1 as an example. In fig. C.1, G_0, G_1, G_2, G_3 , and G_4 are 5 guiding points, B_1, B_2, B_3 are the pieces of B-spline function generated by these guiding points, and P_1, P_2, P_3 are the original parabolas. It is well known that the quadratic spline with guiding points G_1 , G_2, and G_3 is the parabola passing through the middle points M_2 and M_3 of G_1G_2 and G_2G_3 and tangent to them (see Fig. C.1). Moreover, let the original parabola P_1 and P_2 be tangent to G_1G_2 and G_2G_3 at knot points Q_1 and Q_2. We will first show that the maximum discrepancy between P_2 and B_2 do not occur between M_2 and Q_2 or Q_3 and M_3.

<u>Lemma C.1</u>. In Figure C.2, AB is a straight line. P_1 and P_2 are two parabolas intersecting at O and tangent to AB at A and B respectively. AC denote the distance from A to P_2 and BD denotes the distance from B to P_1. Then the Maximum distance between P_1 and P_2 in ACBD is smaller than max(AC, BD).

<u>Proof</u>. Connect B and C. By the convexity of a parabola we see that AC is larger than any distance on AO to P_2 . Similarly one can show BD is the maximum distance in the DOB section. The lemma is proven.

Now go back to Fig. C.1. We now know that the maximum distance between B_2 and P_2 occurs either at the boundary points Q_2 and Q_3 or at a possible local maximum between Q_2 and Q_3 . (Note: The maximum can occur between M_2 and M_3or one M and one Q depending on the positions of M's and Q's. In Fig. C.1, the maximum occurs between the two Q's.) To find the distance between a point and a parabola has been discussed in Appendix B. To find the local maximum between two parabolas, one can transfer one of the two parabolas into the standard form. Let the two parabolas be

$$y=x^2$$
$$f(x,y)=(x+By)^2 + Dx + Ey + F=0.$$

Suppose that in Fig. C.2 P_1 is the standardized parabola and P_2 is $f(x,y)=0$. Then by the convexity of P_1 and P_2 it can be easily seen that the distance between any point on the EG section of P_2 and P_1 is no larger than max($\overline{EF}$, $\overline{GF}$). Naturally, we choose F to be the point so that EF has the (minimum) distance between P_1 and P_2 at E by the method described in Appendix B. Thus, the distance between P_1 and P_2 can be estimated to any accuracy by increasing the number of divisions on P_2 .

Appendix D. Add one guiding point to existing guiding points

Take Figure C.1 again as an example. The reason that there is a discrepancy between B_2 and P_2 is because Q_2 , M_2 and/or Q_3 ,M_3 do not coincide. If $M_2=Q_2$, and $M_3=Q_3$, then there would be no discrepancy. One way to reduce the distance between M_2 and Q_2 or M_3 and Q_3 is to add more guiding points for G_1 , G_2, or G_3 . For example, if G_1 is replaced by two guiding points G_{11} and G_{12}, then the distance between Q_2 and the new midpoint M_2^* will be smaller than M_2Q_2 (see Figure D.1). Thus the fit is better. In general suppose n guiding points are needed for a parabola $y=x^2$ with knots at (a, a^2), and (b, b^2), with a<b. Then the n guiding points should be the intersections of the n+1 tangents at (u_i,u_i^2), i=0,1,2,...,n, where

$$u_i = a + \Delta i \quad , \quad \Delta = (b-a)/n. \qquad (D.1)$$

The guiding points for a general parabola (6) can be found by (D.1) and transformation in Appendix B. To prove that the guiding points constructed by the above procedure actually generate the original parabola, we let the intersections of the two tangents at (u_{i-1} , u_{i-1}^2), (u_i ,u_i^2) be G_i=(x_i ,y_i). Then it is straightforward to show that the midpoint requirements

$$x_i + x_{i+1} - 2u_i^2 = 0$$
$$y_i + y_{i+1} - 2u_i^2 = 0, \quad i=1,2,\ldots,n-1,$$

and the slope requirements

$$y_{i+1} - y_i - 2u_i(x_{i+1} - x_i) = 0, \quad i=0,1,\ldots,n-1,$$

are satisfied.

The convergence of this procedure is not difficult to show because the guiding points in the above procedure produce a perfect fit except at the junctions of two parabolas. As the number of guiding points increases, the distance between the two neighboring guiding points at the two sides of a knot point decreases. It can be seen in Figure C.1 that the maximum discrepancy between B_2 and P_2 cannot be larger than the distance from G_2 to the base M_2M_3 of the triangle $M_2 G_2 M_3$. As the number of guiding points tends to infinite, the distance between neighboring guiding points tends to 0 and so does the maximum discrepancy.

Appendix E. Finding the intersections of two B-splines

Suppose curve C_1 is guided by G_1, G_2, G_3 and curve C_2 is guided by G_1', G_2', G_3'. Let $A=(G_1 + G_2)/2$, $B=(G_2 + G_3)/2$, $A'=(G_1' + G_2')/2$, and $B'=(G_2' + G_3')/2$. Then C_1 and C_2 will not intersect if

$$\begin{aligned} &\max(A, P, B)x < \min(A', P', B')x \\ \text{or } &\min(A, P, B)x > \max(A', P', B')x \\ \text{or } &\max(A, P, B)y < \min(A', P', B')y \\ \text{or } &\min(A, P, B)y > \max(A', P', B')y\,. \end{aligned} \qquad (E.1)$$

where the subscripts x and y denote the x- and y- coordinates of the points. The reason for doing (E.1) is because it requires little computing time. Any one of the conditions in (E.1) implies that triangles APB and A'P'B' do not intersect. Either do C_1 and C_2. If none of the four conditions is satisfied, then we need to find the possible intersections of C_1 and C_2 by algebraic method. Let the functions of C_1 and C_2, described in (4), be

$$C_1 : a_1u^2 + b_1u + c_1$$
$$C_2 : a_2u^2 + b_2u + c_2$$

where a_i, b_i, and c_i, are vectors. Then for any solution of

$$a_1 u_1^2 + b_1u_1 + c_1 = a_2 u_2^2 + b_2u_2 + c_2 \qquad \text{(E.2)}$$

with $0 \leqslant u_1 \leqslant 1$, and $0 \leqslant u_2 \leqslant 1$, it is an intersection of C_1 and C_2. Which segment of the curve is hidden and should be removed from the display is determined by the physical position of the two elements specified by the animator. In general (E.2) turns out to be a quartic equation which can be solved by a standard formula. The formula, although computationally simple, is too cumbersome to be listed here. It can be found in most algebra book or in the CRC Standard Tables.

Appendix F. Matching of two B-splines with unequal numbers of guiding points

Suppose the first curve is controlled by guiding points $G_1, G_2, \ldots, G_m$ and the second curve is controlled by $G'_1, G'_2, \ldots, G'_m$ and G_1 is corresponding to G'_1 and G_m to G'_m. There seems no way to make some one point to multiple points correspondence without introducing some discontinuity, nor does there seem to be a way to force $m=m'$ by adding or reducing the number of guiding points without producing some discrepancy. Our method is first compute the total length of the two curves and then make a one to one correspondence proportionally to their relative positions in the curve. Fortunately, the computation is still quite simple. Since the curves are piecewise parabolas, we may represent each piece by

$$P(u) = \binom{a_x}{a_y}u + 2\binom{b_x}{b_y}u + \binom{c_x}{c_y}, \quad 0 < u < 1 .$$

Thus the length of P(u) is

$$L = \int \{(2a_x u + b_x)^2 + (2a_y u + b_y .)^2\}^{1/2} du$$

$$= a\{ (1+b)\sqrt{(1+b)^2 + c^2} - b\sqrt{a^2 + c^2} + c \ln \frac{1+b + \sqrt{(1+b)^2 + c^2}}{b + \sqrt{b^2 + c^2}} \}$$

with

$$a = \sqrt{a_x^2 + b_y^2}$$

$$b = \frac{a_x b_x + a_y b_y}{a^2}$$

$$c = (a_x b_y - a_y b_x)^2 / a^4 .$$

Let the length of the parabola guided by $P_{i-1} P_i P_{i+1}$ be L_i and P'_{i-1} P'_i P'_{i+1} be L'_i . Then the lengths of C_1 and C_2 are respectively,

$$L_1 = L_2 + L_3 + \ldots + L_{m-1} ,$$

$$L = L'_2 + L'_3 + \ldots + L'_{m-1} .$$

Thus, the point on C_1 which is ℓ from its starting point $(P_1 + P_2)/2$ will be assigned to the point $\ell L'/L$ distance away from its starting point $(P'_1 + P'_2)/2$.

References

1. D. A. Ballard and C. M. Brown, Computer Vision, Prentice Hall, 1982.

2. F. L. Bookstein, Fitting conic section to scattered data, Computer Graphics and Image Processing, 9, 1979, 56-71.

3. K. N. Burtnyk and M. Wein, Computer Animation, Encyclopedia of Computer Science and Technology, Vol. 5, 1977, Pub. Marcel Dekker, N. Y., 397-436.

4. E. Catmull, The problems of computer assisted animation, Computer Graphics, 12(3),1980, 6-10.

5. G. Chaikin, An algorithm for high speed curve generation, Computer Graphics and Image Processing, 3, 1974, 346-349.

6. R. J. Hackathorn, ANIMA II: A 3-D Color animation system, Tutorial and selected readings in Interactive Computer Graphics,COMPCON 80, Edited by H. Freeman, IEEE Computer Society, 1980, 364-374.

7. J. Korein and N. Badler, Temporal and anti-aliasing in computer generated animation, Computer Graphics, 17(3),1983, 377-388.

8. Y. Z. Liao, A two-stage method of fitting conic arcs and straight-line segments to digitized contours, In Proc. IEEE Pattern Recognition and Image Processing Conference (Dallas, Texas, 1981), 224-229.

9. J. J. Little and T. K. Peucker, A recursive procedure for finding the intersection of two digital curves, 1979 CGIP 10, 159-171.

10. O. Lozover and K. Preiss, Automatic generation of cubic B-spline representation for a general digitized curve, Eurographics, J. L. Encarnacao ed., 1981, 119-126.

11. F. E. Parke, Computer generated animation of faces, Master's thesis, University of Utah, Salt Lake City, UTEC-CSC-72-120, June 1972.

12. T. Pavlidis, Algorithms for Graphics and Image Processing, Computer Science Press, 1982.

13. T. Pavlidis, Curve fitting with conic spline, ACM Transcation on Graphics, 2, 1983, 1-31.

14. R. F. Riesenfeld, Application of B-spline approximation to geometric problems of computer-aided design, Ph.D. Thesis, Syracuse University, 1973.

15. R. F. Riesenfeld, On Chaikin's algorithm, Computer Graphics and Image Processing, 4, 1975, 304-310.

16. S. C. Wu, J. F. Abel, and D. P. Greenberg, An interactive computer graphics approach to surface representation, Computer Graphics and Image Processing, 1977, 703-712.

17. F. Yamaguchi, A new curve fitting method using a CAT computer display, Computer Graphics and Image Processing, 1978, 425-437.

Table 1. The Initial Knots, Slopes, and Guiding Pionts in Figure 1

Point	Q	Slope	G (i,1)
1	(9.50,8.90)	-0.649	(8.30,9.60)
2	(10.1,6.50)	0.300	(12.0,7.20)
3	(8.50,3.70)	-5.000	(8.00,6.00)
4	(6.85,2.10)	-0.500	(9.00,1.00)
5	(5.50,5.50)	8.500	(5.00,3.00)
6	(4.10,6.20)	0.152	(5.40,6.40)
7	(1.90,7.30)	1.467	(0.80,5.70)
8	(1.85,8.70)	-1.640	(2.30,7.90)
9	(4.20,10.1)	-0.411	(-0.2,2.00)
10	(5.60,11.2)	6.438	(5.40,9.70)
11	(8.10,11.3)	8.000	(7.00,2.00)

Table 2. The First Three Iterations in Fitting Figure 1 by a B-spline

Parabola	1st iteration		2nd iteration		3rd iteration	
	G. P.	Disc.	G. P.	Disc.	G. P.	Disc.
1	(8.3,9.6)	0.828	No Change	0.166	No change	No change
2	(12.,7.0)	0.126	--	--	--	--
3	(8.0,6.0)	0.126	--	--	--	--
4	(9.0,1.0)	0.051	--	--	--	--
5	(5.0,3.0)	0.251	--	--	--	--
6	(5.4,6.49)	0.306	--	--	--	--
7	(0.8,5.7)	0.347	--	--	--	--
8	(2.3,7.9)	0.258	--	--	--	0.137
9	(-2.,12)	0.870	--	--	(0.89,10.3) (4.20,0.10)	0.062
10	(5.4,9.7)	0.757	--	0.574	--	0.240
11	(7.0,20.)	1.696	(6.30,15.7) (8.10,11.3)	0.051	--	--

Table 3. The Last Two Iterations in Fitting Figure 1 by a B-spline

Parabola	7th iteration		8th iteration	
	G. P.	Disc.	G. P.	Disc.
1	(8.3,9.6)	0.158	No change	No change
2	(12.,7.0)	0.124	--	--
3	(8.0,6.0)	0.126	--	0.123
4	(9.0,1.0)	0.206	(8.76,2.32) (6.75,2.10)	0.048
5	(5.97,2.49) (5.50,5.50)	0.107	--	0.105
6	(5.4,6.49)	0.141	--	--
7	(2.45,5.95) (1.90,7.39)	0.015	--	--
8	(2.3,7.9)	0.076	--	--
9	(0.89,10.3) (4.20,0.10)	0.062	--	--
10	(5.4,9.7)	0.169	--	--
11	(5.94,13.4) (8.10,11.3) (6.28,14.5) (6.61,15.7)	0.015	--	--

Table 4. Implementation Information for Figure 6

Curve	Guiding Points	Motion
bowl	21	none
body	11	shifting (hopping), shape change
head	5	rotation (eating)
mouth	2	shape change
ear (left)	8	shape change
(right)	7	shape change
leg (2 front)	6, 5	shifting and shape change
(2 rear)	8, 5	shifting and shape change
tail	5	rotation
eye	4	none

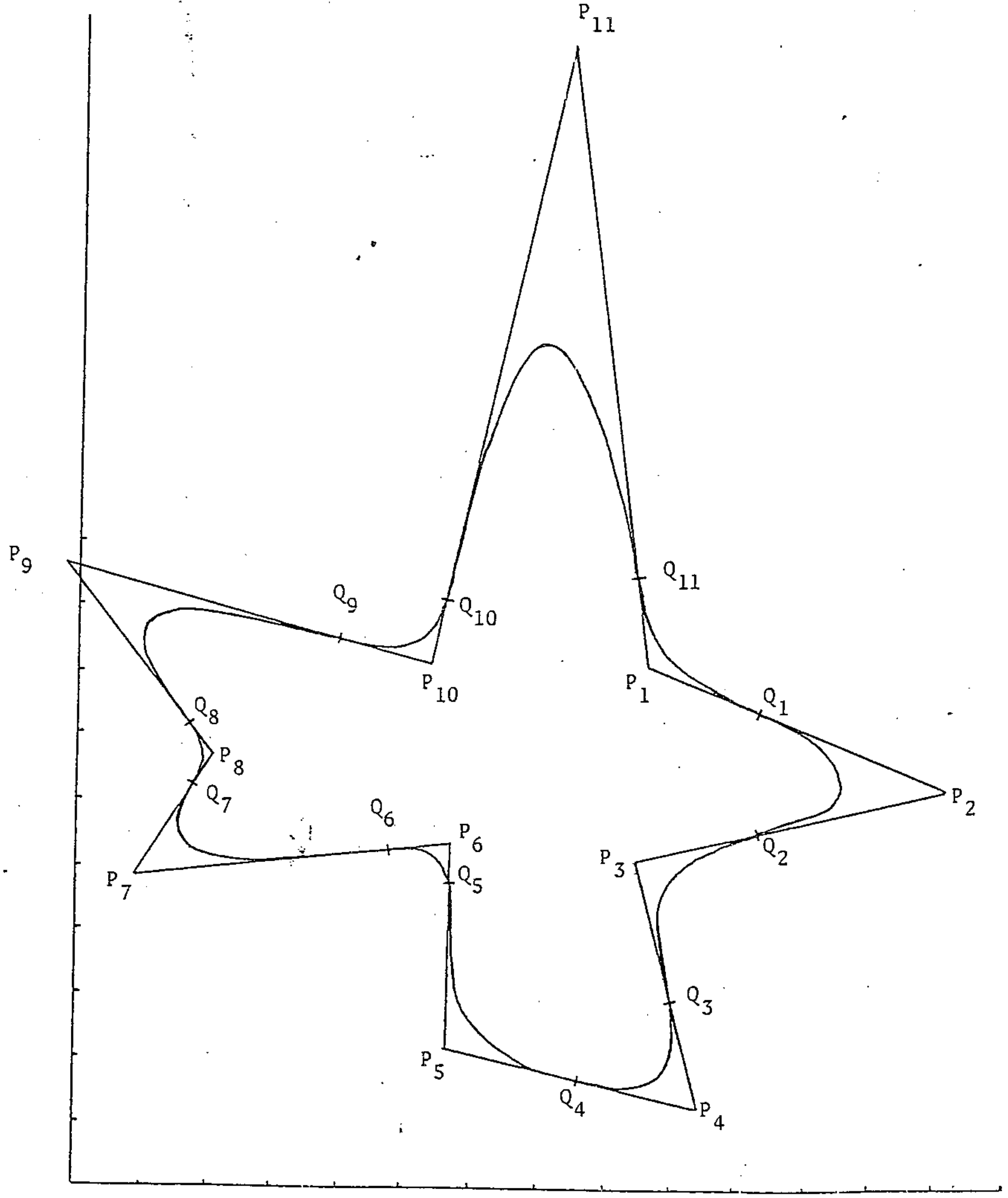


Figure 1. A Segmentation of the Original Curve by Parabolas.

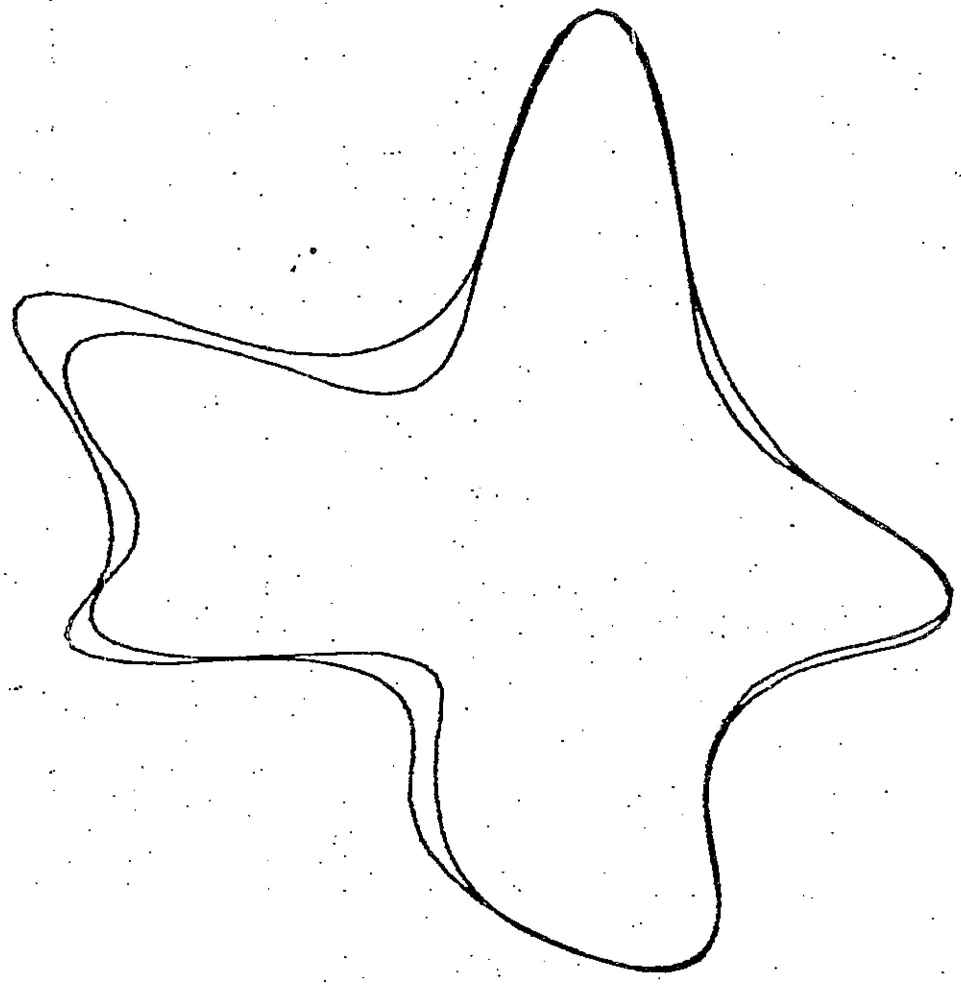


Figure 2. B-spline Fitting with Discrepancy <1.0.
Arrow signs show the original curve and the other is the fitted curve by 12 guiding points.

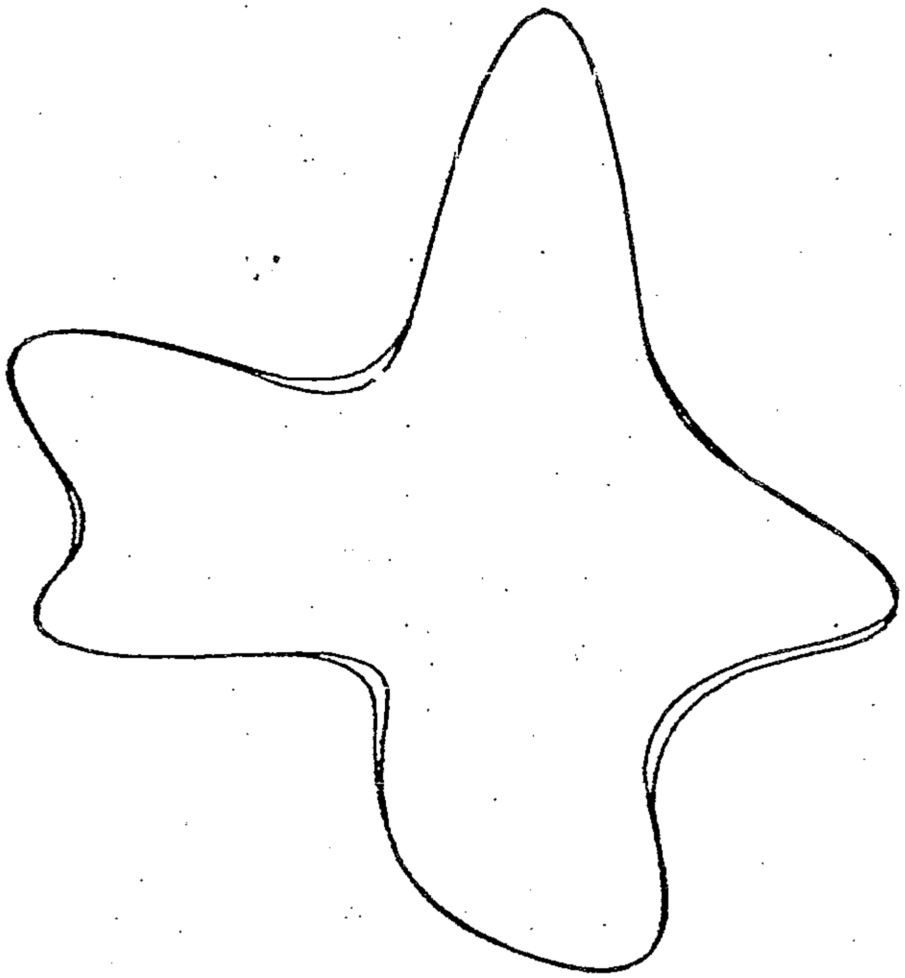
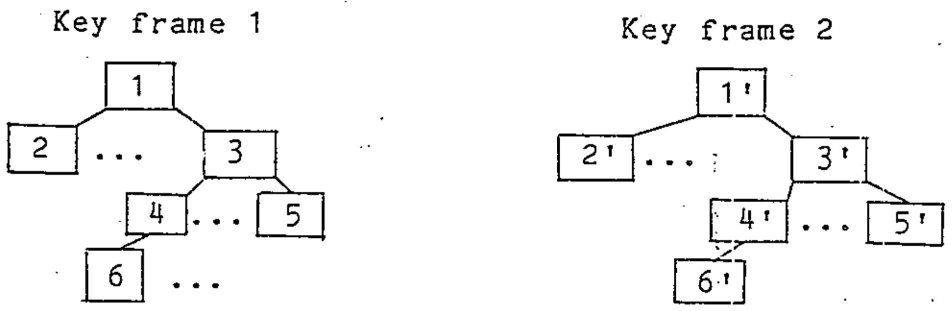


Figure 3. B-spline Fitting with Discrepancy <0.2.



1	1'	:	center of the car
2	2'	:	center of wheel 1
3	3'	:	center of the dog
4	4'	:	center of one of the dog's legs
5	5'	:	center of the dog tail
6	6'	:	center of the foot on leg 4, 4'

Figure 4. Hierarchical Structure of a Picture with Motion

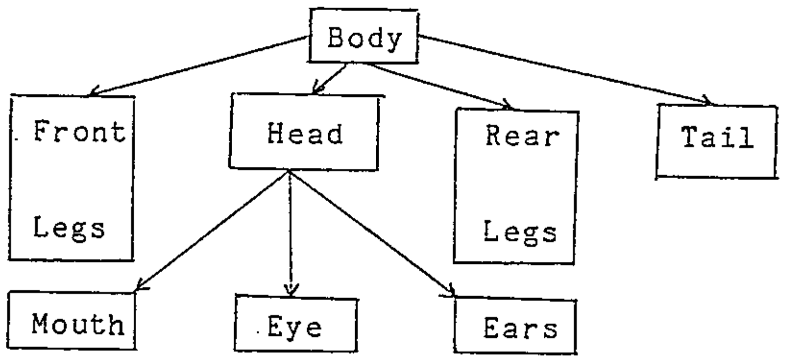
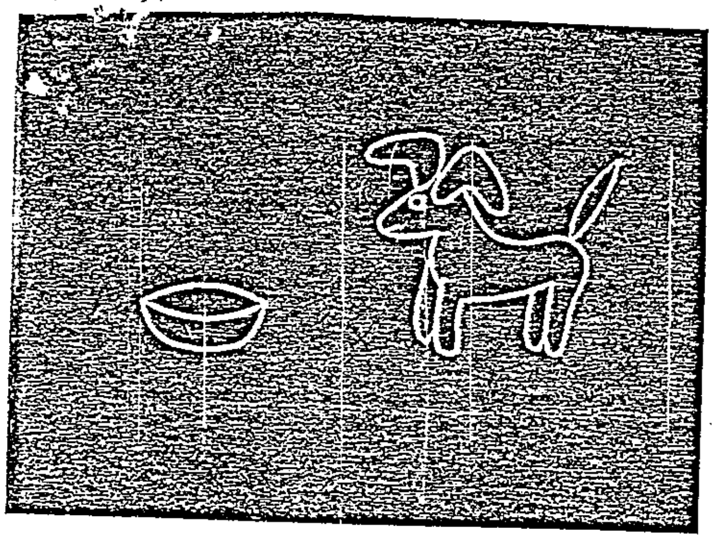
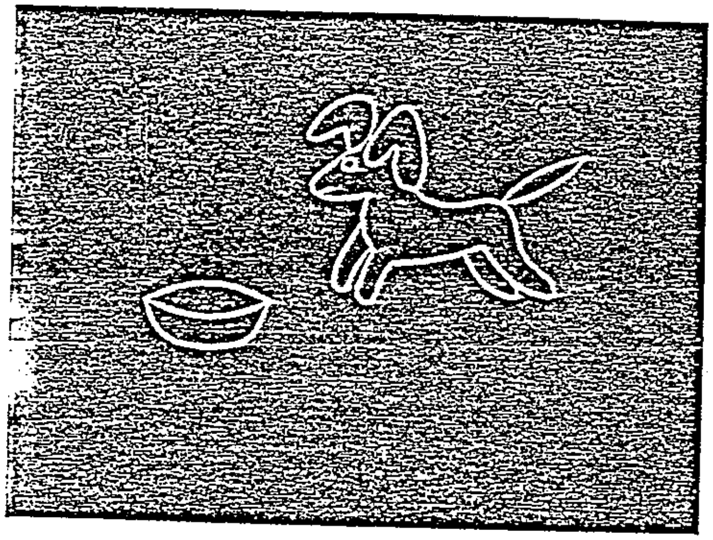
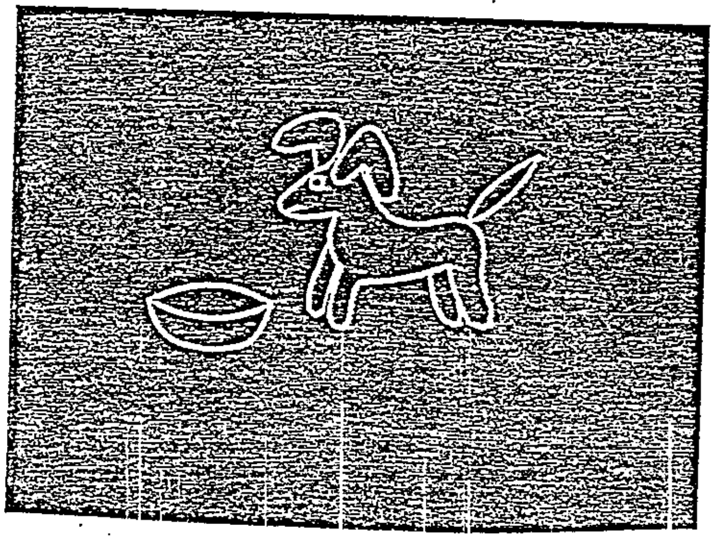
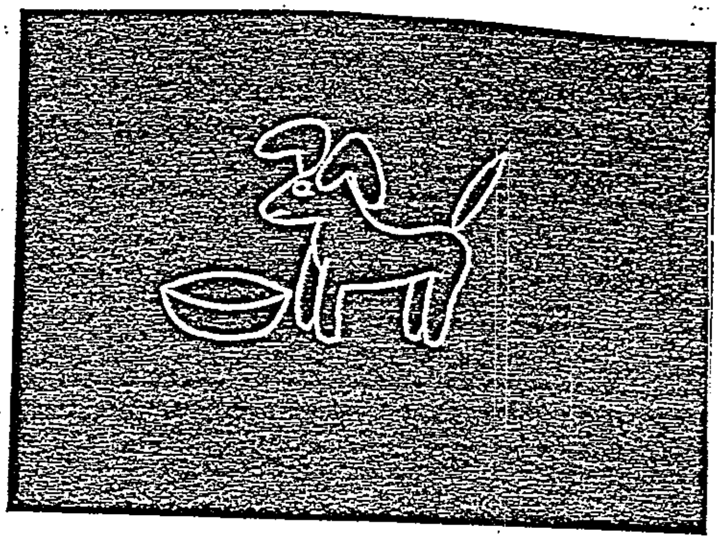


Figure 6. The Hierarchical Data Structure for the Dog in Figure 5


(1)


(2)


(3)


(4)

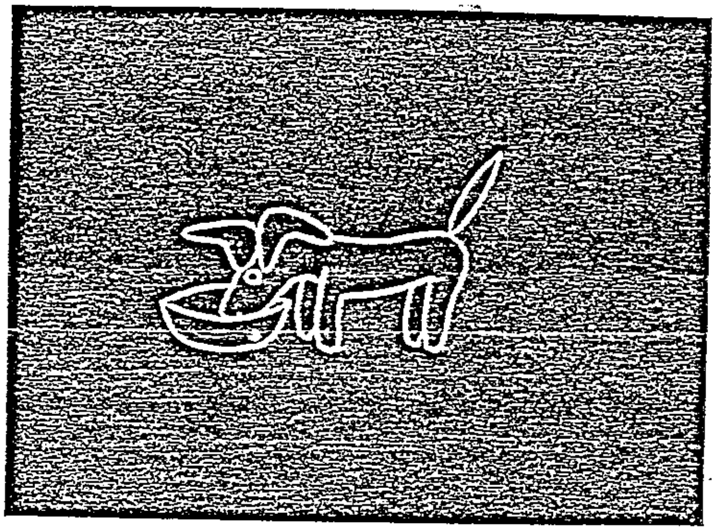

(5)

Figure 5. A sequence of animation pictures for the demonstration of the algorithms described in section 4.

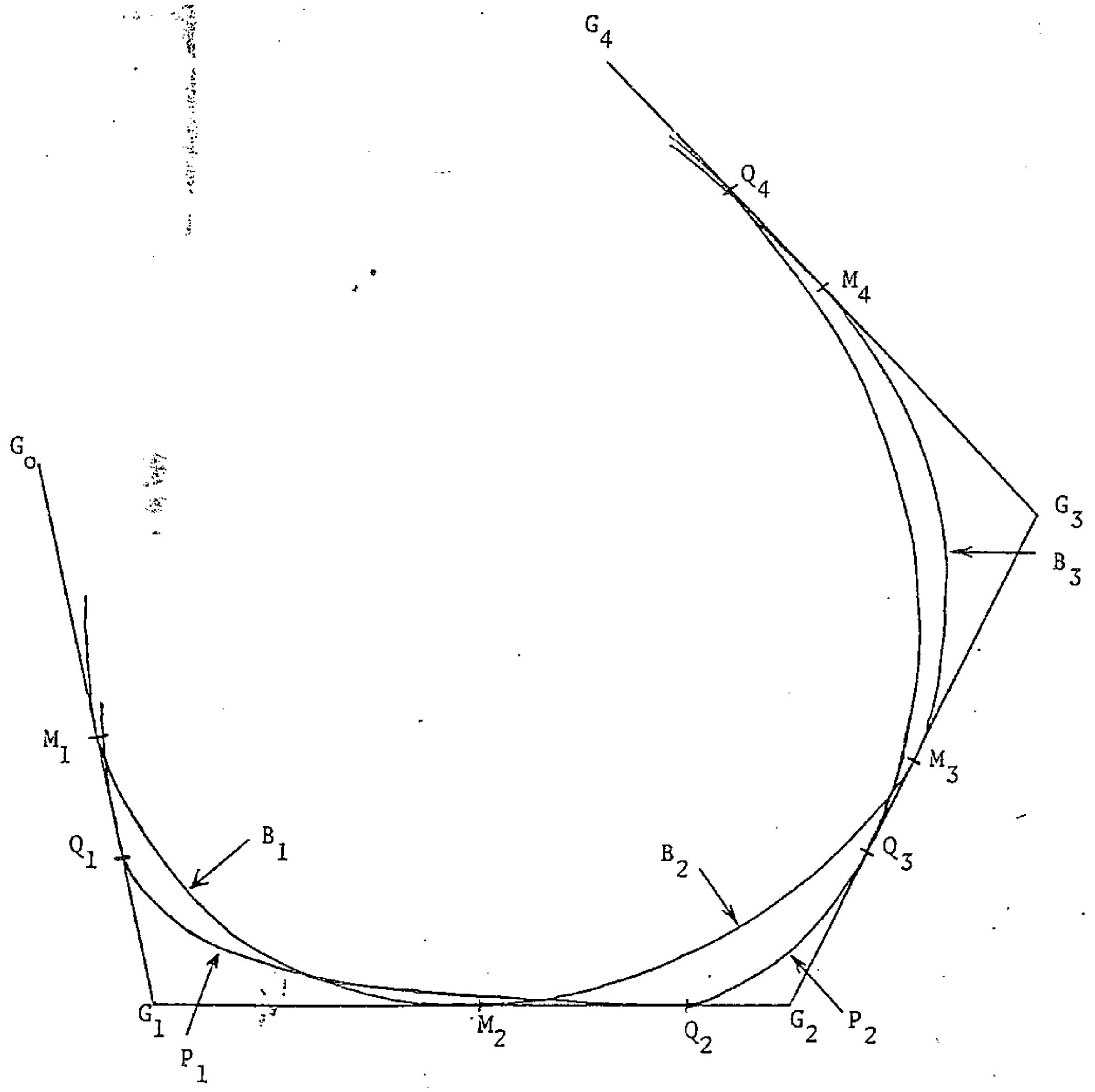


Figure C.1 Adding a Guiding Point to Existing Guiding Points.